1.3 — Introduction to objects and variables

ALEX[1] NOVEMBER 14, 2023

Data and values

In lesson 1.1 -- Statements and the structure of a program (https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/)[2], you learned that the majority of instructions in a program are statements, and that functions are groups of statements that execute sequentially. The statements inside the function perform actions that (hopefully) generate whatever result the program was designed to produce.

But how do programs actually produce results? They do so by manipulating (reading, changing, and writing) data. In computing, **data** is any information that can be moved, processed, or stored by a computer.

Key insight

Programs are collections of instructions that manipulate data to produce a desired result.

A program can acquire data to work with in many ways: from a file or database, over a network, from the user providing input on a keyboard, or from the programmer putting data directly into the source code of the program itself. In the “Hello world” program from the aforementioned lesson, the text “Hello world!” was inserted directly into the source code of the program, providing data for the program to use. The program then manipulates this data by sending it to the monitor to be displayed.

Data on a computer is typically stored in a format that is efficient for storage or processing (and is thus not human readable). Thus, when the “Hello World” program is compiled, the text “Hello world!” is converted into a more efficient format for the program to use (binary, which we’ll discuss in a future lesson).

A single piece of data is called a **value**. Common examples of values include letters (e.g. `a`), numbers (e.g. `5`), and text (e.g. `Hello`).

Random Access Memory

The main memory in a computer is called **Random Access Memory** (often called **RAM** for short). When we run a program, the operating system loads the program into RAM. Any data that is hardcoded into the program itself (e.g. text such as “Hello, world!”) is loaded at this point.

The operating system also reserves some additional RAM for the program to use while it is running. Common uses for this memory are to store values entered by the user, to store data read in from a file or network, or to store values calculated while the program is running (e.g. the sum of two values) so they can be used again later.

You can think of RAM as a series of numbered boxes that can be used to store data while the program is running.

In some older programming languages (like Applesoft BASIC), you could directly access these boxes (e.g. you could write a statement to "go get the value stored in mailbox number 7532").

Objects and variables

In C++, direct memory access is discouraged. Instead, we access memory indirectly through an object. An **object** is a region of storage (usually memory) that can store a value, and has other associated properties (that we'll cover in future lessons). How the compiler and operating system work to assign memory to objects is beyond the scope of this lesson. But the key point here is that rather than say "go get the value stored in mailbox number 7532", we can say, "go get the value stored by this object". This means we can focus on using objects to store and retrieve values, and not have to worry about where in memory those objects are actually being placed.

Although objects in C++ can be unnamed (anonymous), more often we name our objects using an identifier. An object with a name is called a **variable**.

Key insight

An object is used to store a value in memory. A variable is an object that has a name (identifier).

Naming our objects let us refer to them again later in the program.

Nomenclature

In general programming, the term *object* typically refers to an unnamed object in memory, a variable, or a function. In C++, the term *object* has a narrower definition that excludes functions.

Variable instantiation

In order to create a variable, we use a special kind of declaration statement called a **definition** (we'll clarify the difference between a declaration and definition later).

Here's an example of defining a variable named `x`:

```
int x; // define a variable named x, of type int
```

At compile time, when the compiler sees this statement, it makes a note to itself that we are defining a variable, giving it the name `x`, and that it is of type `int` (more on types in a moment).

From that point forward (with some limitations that we'll talk about in a future lesson), whenever the compiler sees the identifier `x`, it will know that we're referencing this variable.

When the program is run (called **runtime**), the variable will be instantiated. **Instantiation** is a fancy word that means the object will be created and assigned a memory address. Variables must be instantiated before they can be used to store values. For the sake of example, let's say that variable *x* is instantiated at memory location 140. Whenever the program uses variable x, it will access the value in memory location 140. An instantiated object is sometimes called an **instance**.

Data types

So far, we've covered that variables are a named region of storage that can store a data value (how exactly data is stored is a topic for a future lesson). A **data type** (more commonly just called a **type**) tells the compiler what type of value (e.g. a number, a letter, text, etc…) the variable will store.

In the above example, our variable `x` was given type `int`, which means variable x will represent an integer value. An **integer** is a number that can be written without a fractional component, such as `4`, `27`, `0`, `-2`, or `-12`. For short, we can say that `x` is an `integer variable`.

In C++, the type of a variable must be known at **compile-time** (when the program is compiled), and that type can not be changed without recompiling the program. This means an integer variable can only hold integer values. If you want to store some other kind of value, you'll need to use a different type.

Integers are just one of many types that C++ supports out of the box. For illustrative purposes, here's another example of defining a variable using data type `double`:

```
double width; // define a variable named width, of type double
```

C++ also allows you to create your own custom types. This is something we'll do a lot of in future lessons, and it's part of what makes C++ powerful.

For these introductory chapters, we'll stick with integer variables because they are conceptually simple, but we'll explore many of the other types C++ has to offer (including `double`) soon.

Defining multiple variables

It is possible to define multiple variables *of the same type* in a single statement by separating the names with a comma. The following 2 snippets of code are effectively the same:

```
int a;
int b;
```

is the same as:

```cpp
int a, b;
```

When defining multiple variables this way, there are two common mistakes that new programmers tend to make (neither serious, since the compiler will catch these and ask you to fix them):

The first mistake is giving each variable a type when defining variables in sequence.

```cpp
int a, int b; // wrong (compiler error)

int a, b; // correct
```

The second mistake is to try to define variables of different types in the same statement, which is not allowed. Variables of different types must be defined in separate statements.

```cpp
int a, double b; // wrong (compiler error)

int a; double b; // correct (but not recommended)

// correct and recommended (easier to read)
int a;
double b;
```

Best practice

Although the language allows you to do so, avoid defining multiple variables of the same type in a single statement. Instead, define each variable in a separate statement on its own line (and then use a single-line comment to document what it is used for).

Summary

In C++, we use objects to access memory. A named object is called a variable. Variables have an identifier, a type, and a value (and some other attributes that aren't relevant here). A variable's type is used to determine how the value in memory should be interpreted.

In the next lesson, we'll look at how to give values to our variables and how to actually use them.

Quiz time

Question #1

What is data?

Hide Solution (javascript:void(0))[3]

> Data is any information that can be moved, processed, or stored by a computer.

Question #2

What is a value?

Hide Solution (javascript:void(0))[3]

> A value is a letter (e.g. `a`), number (e.g. `5`), text (e.g. `Hello`), or instance of some other useful concept that can be represented as data.

Question #3

What is a variable?

Show Solution (javascript:void(0))[3]

Question #4

What is an identifier?

Show Solution (javascript:void(0))[3]

Question #5

What is a type?

Show Solution (javascript:void(0))[3]

Question #6

What is an integer?

Show Solution (javascript:void(0))[3]

Next lesson
1.4 Variable assignment and initialization

4

Back to table of contents

5

Previous lesson
1.2 Comments

6

7

B U URL INLINE CODE C++ CODE BLOCK HELP!

Leave a comment...

Name*

Email*

Notify me about replies:

POST COMMENT

Find a mistake? Leave a comment above!

Avatars from https://gravatar.com/[9] are connected to your provided email address.

585 COMMENTS Newest

Rez
April 8, 2024 2:11 am

I've been programming professionally for over a decade. I was looking for a good resource to recommend to my niece, and your tutorials are so well-written that I suddenly realized I was reading the whole thing! and I'm really enjoying it!

This is a fantastic resource! It's by far the best I've found for people of all skill levels, especially beginners, to learn programming in general, not just C++.

Last edited 3 days ago by Rez

0 Reply

Alex Author
Reply to Rez[10] April 9, 2024 8:07 am

Thanks for the kind words! I appreciate it.

1 Reply

yuunp
February 19, 2024 1:43 am

I dont get this part from this lesson;

When the program is run (called runtime), the variable will be instantiated. Instantiation is a fancy word that means the object will be created and assigned a memory address. Variables must be instantiated before they can be used to store values. For the sake of example, let's say that variable x is instantiated at memory location 140. Whenever the program uses variable x, it will access the value in memory location 140. An instantiated object is sometimes called an instance.

Specifically "Variables must be instantiated before they can be used to store values", so if int x; is decleration, then is it also called instantiation? Since it says that you must instantiate first before storing it some value??

(Srry for my bad grammar)

0 Reply

ddwwwwwwwwww

Reply to yuunp 11 March 24, 2024 7:45 pm

am i the only one that feels like im not getting any of this or

1 Reply

Alex Author

Reply to yuunp 11 February 19, 2024 8:34 pm

`int x` is both a declaration (as it tells the compiler about the type and name of a variable) as well as a definition (it instructs the compiler to write machine code that will cause this variable to be created when the program is run).

When the program is run, variable `x` is instantiated (given a memory address). Once `x` is instantiated, then a value can then be stored in the memory given to that variable.

8 Reply

christopher

Reply to Alex 12 March 6, 2024 9:45 pm

So, declaration introduces the variable with its type and name, instantiation allocates memory for it, and initialization initializes it with a specific value?

1 Reply

Alex Author

Reply to christopher 13 March 10, 2024 2:45 pm

Yes.

3 Reply

Paster
February 12, 2024 2:16 am

Is it better to define variables in main function or in a namespace?

0 Reply

Alex Author
Reply to Paster 14 February 15, 2024 12:25 pm

Variables defined in `main()` are local variables, and can only be used in `main()`. This is the preferable choice if it meets your needs.

Variables defined in a namespace are global variables. They can be used throughout the entire file or program. This is okay for constant variables you use throughout your program, but should generally be avoided otherwise.

0 Reply

Gustavo Nunes Pereira
Reply to Paster 14 February 12, 2024 5:22 pm

It depends on the purpose of these variables you're trying to define. In general, if you are creating a namespace, it is likely that this namespace will be used in other places around your project.If that's your case, maybe it's better to define then inside your namespace. Otherwise, keep them in your main function.

1 Reply

IceFloe
February 11, 2024 5:50 pm

1. can I store multiple values of the same type in one variable?
2. is there a type of variable that can store any value?
3. Does the type of variable depend on how much memory it will occupy in RAM
4. How does the Cache differ from RAM?

1 Reply

Gustavo Nunes Pereira
Reply to IceFloe 15 February 12, 2024 5:39 pm

Hello,

1 - can I store multiple values of the same type in one variable?
R: Yes, there is a data type called array (Vectors/Matrices). This kind of data allows multiple values of the same type to be stored.

2 - is there a type of variable that can store any value?
R: Based on my limited C++ knowledge, there is no data type that allows this behavior.In most recent C++ updates, a new keyword called 'auto' was introduced.This keyword allows you to define a variable without explicitly specifying its data type (only during development time).However, during compile time, the compiler will analyze the value stored in it and set its data type accordingly. Once set, the data type cannot be changed.

Example:

#include <iostream>

int main() {

int a = 26; // Integer data type
auto b = 35; // Integer data type as well, but it will be set during compile time based on it's value.

return 0;
}

3 - Does the type of variable depend on how much memory it will occupy in RAM
R: Not sure about your question, but, every data type has an amount of memory to store its values.

4 - How does the Cache differ from RAM?
R: In this link there's a good explanation about your question: Difference Between Ram and Cache (https://www.geeksforgeeks.org/difference-between-ram-and-cache/)[16]

Last edited 1 month ago by Gustavo Nunes Pereira

0 Reply

IceFloe
Reply to Gustavo Nunes Pereira[17] February 14, 2024 1:00 am

thanks for the answers, I have re-verified the information and you were right,

0 Reply

Mykolas
January 22, 2024 3:36 am

If I have a lot of variables, is it still a better practice to use a different statement for initializing all of them or is that not worth it for the place it will take up?

0 Reply

Alex Author
Reply to Mykolas[18] January 23, 2024 10:10 am

I think so, but if you disagree you can do otherwise.

0 Reply

Brandon
January 16, 2024 12:53 pm

Int score = 10;
(Data Type) (identifier) = (value);
score as a whole is a Variable (a named object)

Last edited 2 months ago by Brandon

5 Reply

Omri
January 15, 2024 9:37 pm

this site is so good so far!!

2 Reply

everyone
Reply to Omri 19 February 3, 2024 3:05 pm

Agreed! It simplifies everything and explains it in plain-speak. Makes entry very easy to grasp & understand.

0 Reply

Abhey
January 12, 2024 6:12 am

the bit that im a little confused about is here a 'variable is being categorized as an object' and as we know in oop the term 'object' is an 'instance of a class' does that mean variable is just a special case for an instance of a class or does the word object holds completely different meaning in these two cases

1 Reply

Alex Author
Reply to Abhey 20 January 13, 2024 5:04 pm

The word object has different meanings in these two cases. It's an unfortunate overload of the word that causes confusion.

7 Reply

Vlad
January 11, 2024 9:23 am

Is it right to consider (int {2}) to be an unnamed object in the statement (int {2};)? Or is it just a value?

Last edited 3 months ago by Vlad

0 Reply

Alex Author
Reply to Vlad 21 January 11, 2024 9:49 pm

Yes, this is a list initialization of an unnamed temporary object.

2 Reply

Cobalt
January 7, 2024 1:02 pm

for anyone else who was a little confused on how ram works (like me), every time you run a program, the memory starts from a clean slate. so every time you run, the variable is allocated to a new mailbox and isn't a persisting piece of data stored on your computer.

I think!

2 Reply

AbeerOrTwo
December 13, 2023 7:35 pm

love me a good integer

3 Reply

Te0_faN
December 10, 2023 6:01 am

```
#include <iostream>

   int main()
  { std::cout<<"Nice lesson!";
    return 0;
  }
```

10 Reply

voxoking
Reply to Te0_faN March 20, 2024 2:16 pm

I would love to understand this joke

0 Reply

Alex Author
Reply to voxoking March 21, 2024 2:03 pm

It's not a joke. It's just a program that, when compiled and executed, prints "Nice lesson!" to the console.

0 Reply

devLS
November 24, 2023 8:26 am

Great tutorial, I just don't think that beginners should be learning type "double" at all in this. Introduce them to int and possibly unsigned int. But don't mention a thing about double. Otherwise, I think you teach this great!

1 Reply

evan
Reply to devLS February 13, 2024 10:47 am

why? beginners have to learn it eventually. keeping things from beginners will prevent them from being anything other than beginners.

0 Reply

Maximum G
November 1, 2023 10:23 pm

I'm reviewing from the beginning after finishing chapter 15. I don't know why. But with what I have learned, this lesson becomes hard to understand for me.

I can understand programs manipulate data to produce results. But the next section goes straight into how C++ access data from RAM (through an object), without explaining why we need to store data in the first place. The only program we have seen at this point is the "Hello world" one, but it doesn't need to store any data to produce results.

1 Reply

Alex Author
Reply to Maximum G November 3, 2023 8:16 pm

I appreciate the feedback. I added some additional text to the lesson talking about RAM that I hope addresses this. Thoughts?

5 Reply

Maximum G
Reply to Alex November 3, 2023 8:58 pm

Much better! The added section now connects the last and next sections logically.

0 Reply

Favour
October 28, 2023 1:18 pm

What's the difference between a variable and an identifier?

0 Reply

Vinay
Reply to Favour October 30, 2023 4:36 pm

A named "object" is called a "variable" and the name of the object is called the "identifier".

1 Reply

Alex Author
Reply to Favour October 30, 2023 11:57 am

A variable is an object that has an identifier. We use variables to store values in memory. An identifier is the name of something.

Both variables and functions have an identifier:

```cpp
int x; // x is the identifier of this object
void print(); // print is the identifier of this function
```

9 Reply

Favour
Reply to Alex October 30, 2023 4:03 pm

Thanks

0 Reply

David
October 19, 2023 10:45 am

You mentioned An object is a region of storage (usually memory) that can store a value. The instance of a class is also called an object, how do they differ?

0 Reply

Alex Author
Reply to David October 21, 2023 1:05 pm

In non-oop, an object is an instance of a data type.
In OOP, an object is an instance of a class type (class, struct, or union).

The OOP version restricts the definition of object to just class types.

3 Reply

Asmo
Reply to Alex February 13, 2024 6:27 am

can you please explain it further so that i can understand too like what is diff between oop and no-oop ?

0 Reply

Alex Author
Reply to Asmo February 16, 2024 8:52 am

https://www.learncpp.com/cpp-tutorial/introduction-to-object-oriented-programming/

1 Reply

jake from state farm
October 2, 2023 8:11 pm

thank you a lot I leant to declare variables in other websites but this is the first time I actually understood what a variable actually was and why we were declaring it.

1 Reply

Krishnakumar
October 2, 2023 9:58 am

>In C++, direct memory access is discouraged.

discouraged, but possible?

0 Reply

Alex Author
Reply to Krishnakumar October 3, 2023 10:29 am

Yes, I mean, you can reinterpret_cast an integral value into a pointer and then dereference the pointer... but you'll probably get UB as a result.

3 Reply

ether
September 14, 2023 11:05 pm

Cool... Awesome!!!

1 Reply

SillyPersonWhoShouldntBeLearningProgramming
September 9, 2023 4:46 am

So, I am trying to learn programming and I have tried a few different approaches. I have to say that you, Alex, are doing a great job so far. Thank you. A neanderthal like me is very appreciative.

I am taking a Procedural Programming class and it is all based on Visual Basic, of which isn't really helpful (and increasing irrelevant). I downloaded Stroustrup's book, but it is a bit hard to follow, mostly because I hate reading PDF's, and because they created additional support files that I am not trying to pay for and never use again. And I borrowed a library book about UML and C++ that reads like a technical manual. So, this easy to understand, well organized site, seems to mesh well with my experience level and learning style. I'll just everything else as supplementation.

2 Reply

Links

1. https://www.learncpp.com/author/Alex/
2. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/
3. javascript:void(0)
4. https://www.learncpp.com/cpp-tutorial/variable-assignment-and-initialization/
5. https://www.learncpp.com/
6. https://www.learncpp.com/cpp-tutorial/comments/
7. https://www.learncpp.com/introduction-to-objects-and-variables/
8. https://www.learncpp.com/cpp-tutorial/introduction-to-functions/
9. https://gravatar.com/
10. https://www.learncpp.com/cpp-tutorial/introduction-to-objects-and-variables/#comment-595525
11. https://www.learncpp.com/cpp-tutorial/introduction-to-objects-and-variables/#comment-593809
12. https://www.learncpp.com/cpp-tutorial/introduction-to-objects-and-variables/#comment-593835
13. https://www.learncpp.com/cpp-tutorial/introduction-to-objects-and-variables/#comment-594385
14. https://www.learncpp.com/cpp-tutorial/introduction-to-objects-and-variables/#comment-593580
15. https://www.learncpp.com/cpp-tutorial/introduction-to-objects-and-variables/#comment-593544
16. https://www.geeksforgeeks.org/difference-between-ram-and-cache/
17. https://www.learncpp.com/cpp-tutorial/introduction-to-objects-and-variables/#comment-593602
18. https://www.learncpp.com/cpp-tutorial/introduction-to-objects-and-variables/#comment-592677
19. https://www.learncpp.com/cpp-tutorial/introduction-to-objects-and-variables/#comment-592403
20. https://www.learncpp.com/cpp-tutorial/introduction-to-objects-and-variables/#comment-592258
21. https://www.learncpp.com/cpp-tutorial/introduction-to-objects-and-variables/#comment-592187